# Lecture 5

Distributed algorithms

vs.

sublinear algorithms

(on sparse graphs)

max degree $\leq \Delta$

constant!

Today:

a way of designing lots of sublinear algorithms for sparse (ave deg $O(1)$) graphs

idea: use known fast distributed algs

+

reduction

**Vertex Cover:** $V' \subseteq V$ is "vertex cover" (VC) if $\forall (u,v) \in E$, either $u \in V'$ or $v \in V'$

**VC question:** What is min size of VC?

**note:**
- in deg $\leq \Delta$ graph, $|VC| \geq \frac{m}{\Delta} \geq \frac{n \cdot \Delta_{ave} \cdot \frac{1}{2}}{\Delta_{max}}$ since each node covers $\leq \Delta$ edges
- in general VC is NP-complete
- there is poly time 2-approx via matching

Question: Can you approx V.C. in Sublinear time?

multiplicative? no! can't distinguish graph with
0 edges from graph with
1 edge in sublinear time

additive? hard since NP-complete
need some mult error (at least 1.36
maybe factor of 2)

Combination mult + add error?

def $\hat{y}$ is $(\alpha, \varepsilon)$ - approximation of soln value $y$
for minimization problem if

$$y \le \hat{y} \le \alpha \cdot y + \varepsilon$$

(analogous defn for maximization problem)

# Background on LOCAL distributed Algorithms:

well studied model

- Network
  - processors
  - links

  } max degree $\Delta$

- Communication round
  - nodes perform computation on (input bits, history of received msgs, random bits)
  - nodes send msgs to nbrs
  - nodes receive msgs from nbrs

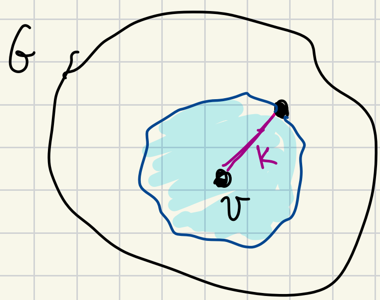
- Vertex cover on a distributed network:
  - network graph = input graph
  - at end, each node knows if it is in VC

    (doesn't necessarily know about other nodes or total size)

# LOCAL Distributed vs. Sublinear time:

main insight:

In $k$-round LOCAL algorithm, output of node $v$ can only depend on nodes of distance $\leq k$ from $v$. At most $\Delta^k$ of these!




$\Rightarrow$ Can sequentially simulate $v$'s view of distributed computation in $\leq \Delta^k$ queries & at end know if $v$ in or out of VC

Comment: if LOCAL algo is randomized, $v$ needs to know random bits of all $\leq \Delta^k$ nbrs
must be consistent

So fast distributed LOCAL for VC

$\Longrightarrow$ $\underbrace{\text{fast}}_{\text{sublinear query?}}$ sequential "oracle" for VC

what about sublinear time?

Is there a fast distributed LOCAL alg for VC?

Yes, will see more soon

What do we do with this oracle?

use to estimate size of VC

via sampling.

# Estimating size of VC

## Parnas-Ron Framework:

Sample nodes of graph $v_1 \cdots v_r$

for each $v_i$

simulate distributed LOCAL algo to see if $v_i \in VC$

Output $\frac{\# \; v_i\text{'s in VC}}{r} \cdot n$

runtime: $O(r \cdot \Delta^{k+1}) \approx O(\frac{1}{\varepsilon^2} \cdot \Delta^{k+1})$

$k$ = # rounds of distr. algo.

$\Delta$ = max degree

Proof of correctness: Chernoff-Hoeffding bnds

# A fast distributed algorithm for VC

idea: primal - dual based approx algo

$\uparrow$ matching $\quad\quad$ $\uparrow$ V.C.

Relate V.C. to dual problem of fractional matching!

What is a fractional matching?

- assign weights $x_e$ in $[0,1]$ to edges $e$

  s.t. $\forall v \quad \sum_{\underbrace{e \ni v}_{\text{edges containing } v \text{ as end pt}}} x_e \le 1$

- value of fract matching $= \sum_e x_e$

in integral matching, $e \in \{0,1\}$ & $\sum_{e \ni v} x_e$ is either $\underset{\text{unmatched}}{0}$ or $\underset{\text{matched}}{1}$

# Algorithm

- Init $x_e = \frac{1}{\Delta}$ $\forall$ edge $e$ ← start with trivial soln to matching (primal)

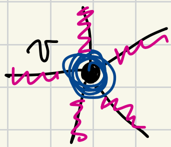
  initially sum around each node $\leq 1$

- For $i = 1$ to $\log_{1+\gamma} \Delta$

  - for each node $v$ s.t. $\sum_{e \ni v} x_e \geq \frac{1}{1+\gamma}$,

    - add $v$ to V.C.
    - freeze $x_e$ $\forall$ $e \ni v$

      ($v$ sends "frozen" msg to all nbrs)




  - for each unfrozen edge $e$, set $x_e \leftarrow (1+\gamma) x_e$

    gradually increase until become maximal

(At end any remaining $v$ are not in V.C.)

#rounds: $1 + \log_{1+\gamma} \Delta$

time to simulate: $\Delta^{\log_{1+\gamma} \Delta}$

Why is it a V.C.?

if e frozen, then $\geq 1$ endpt in V.C., else

if e survives until end, $X_e > (1+\gamma)^{\log_{1+\gamma} \Delta} \cdot \frac{1}{\Delta} = \frac{\Delta}{\Delta}$

$> 1$

$\Rightarrow$ endpts would join V.C.

$\Rightarrow$ no e survives

$\Rightarrow$ at least one endpt in V.C. $\blacktriangle$

Why is it a small V.C.?

Note: our algorithm finds a fractional matching

why? if $v$ doesn't freeze edges,

$$\sum_{e \ni v} x_e < \frac{1}{1+\gamma}$$

so in next round $\sum_{e \ni v} x_e \le \left(\frac{1}{1+\gamma}\right) \cdot (1+\gamma) = 1$

$\Rightarrow$ gets frozen before getting bigger than 1.

integral

plan: show (1) (any) fractional matching value $\le$ min V.C.

(2) (our) Output V.C. $\le 2(1+\gamma) \cdot$ (our) fractional matching value

$\Rightarrow$ Output V.C. $\le 2 \cdot (1+\gamma) \cdot$ min V.C.

in particular, max + the one found in our algorithm

Any fractional matching value $\leq$ min V.C. :

recall: integral max matching $\leq$ min V.C.

since each edge in matching contributes $\geq 1$ node to any V.C.

Given valid fractional matching with value $\sum_e x_e$

+ optimal V.C. $V_{OPT}$

$\forall e$, assign $x_e$ to endpt in $V_{OPT}$ $\Rightarrow$ if both in $V_{OPT}$, choose arbitrarily

total wt assigned to nodes $= \sum_e x_e$

each node in $V_{OPT}$ gets $\leq 1$

since comes from fract. matching

$$\Rightarrow |V_{OPT}| \geq \frac{\text{total wt}}{\text{max wt per node}} \geq \frac{\sum_e x_e}{1} = \sum_e x_e$$

$\blacksquare$

## Output V.C. $\leq 2(1+\gamma)$ (our) Fractional matching value

Given output V.C. $\hat{V}$

for every $u \in \hat{V}$, assign value 1

(so total value $= |\hat{V}|$)

Split u's value among neighboring edges proportional to $X_e$: ← from algorithm

each edge $e'$ gets $\dfrac{X_{e'}}{\sum_{e \ni u} X_e} \leq \dfrac{X_{e'}}{\frac{1}{1+\gamma}} = (1+\gamma) X_{e'}$ from u

can also get at most $(1+\gamma) X_{e'}$ from other endpt

total assigned to $e' \leq 2(1+\gamma) X_{e'}$

$$\text{total value} = |\hat{V}| \leq 2(1+\gamma) \underbrace{\sum_e X_e}_{\text{wt of our fract. matching}} \leq 2(1+\gamma)\,|V_{OPT}|$$

∎